Pacific University
CommonKnowledge

Volume 1 (2001)

Interface: The Journal of Education, Community and Values

11-1-2001

Using PHP for Websites: Creating a Dynamic Interface in Theory and Practice

Jesse Snyder
Pacific University

Using PHP for Websites: Creating a Dynamic Interface in Theory and Practice

Description

Many of you have created web pages. Many have not. This article isn't geared toward either group. It is meant to be an article that might persuade those who haven't created a website to make one, and those who have to further their web skills. If you have any experience with HTML, you are capable of doing what we are going to cover shortly. If not, you might pick something up or might be able to have your IT person help you along the way.

This article is hosted at The Berglund Journal, which has used the methods I will describe below. My hope is that from reading this article, you too can use the power of PHP to make your web building go much faster and be more efficient. This article centers around one of the many possibilities of PHP: designing an interface for your entire site that you can change by only modifying a few files.

I encourage you to read this article, play around with PHP and create. Go check out some PHP books from the library or even buy one. You could also check out www.php.net as it is the best resource for PHP online, and best of all, it's free!

Using PHP for Websites: Creating a Dynamic Interface in Theory and Practice

Posted on **November 1, 2001** by **Editor**

By Jesse Snyder <snyderjw@pacificu.edu>
Senior, Computer Science Major at Pacific University
Webmaster for the Berglund Center for Internet Studies and the Matsushita Center for Electronic Learning

Preface:
Many of you have created web pages. Many have not. This article isn't geared toward either group. It is meant to be an article that might persuade those who haven't created a website to make one, and those who have to further their web skills. If you have any experience with HTML, you are capable of doing what we are going to cover shortly. If not, you might pick something up or might be able to have your IT person help you along the way.

This article is hosted at The Berglund Journal, which has used the methods I will describe below. My hope is that from reading this article, you too can use the power of PHP to make your web building go much faster and be more efficient. This article centers around one of the many possibilities of PHP: designing an interface for your entire site that you can change by only modifying a few files.

I encourage you to read this article, play around with PHP and create. Go check out some PHP books from the library or even buy one. You could also check out www.php.net as it is the best resource for PHP online, and best of all, it's free!

PHP? HTML?
PHP stands for "Personal Homepages" but later was changed to "PHP: Hypertext Processor." According to www.webopedia.com, "PHP Hypertext Preprocessor is a server-side, HTML embedded scripting language used to create dynamic Web pages," and this is exactly what we are going to do with it; create dynamic web pages.

Let's break it down for a moment. HTML is the language that drives the web. HTML or "Hypertext Markup Language" is composed of "tags" that determine what web pages look like. For example, if I wanted to make some bold text in HTML, I would write <B>My Text</B>. Your

web browser of choice would read these bold “tags” surrounding the text and display the text as bold. Pretty simple huh?

All web pages are made up of HTML tags. These tags also have attributes that can change other aspects of your page. You can make pictures display at a certain width or height, change the color of text, or tell a web page where a java applet is located.

Where does PHP fit into all of this? It was noted earlier that PHP is a server-side scripting language. What this means is that when you point your browser at a URL, the server, not the machine you are on, does the processing of the language. This has many advantages. First of all, there is no need for plug-ins. Anyone with a web browser is capable of viewing PHP enabled sites, unlike some sites that might require you to download Flash player (which can only play Flash movies) for example. Second, this is much faster than a client side scripting language. For example, JavaScript is a client side scripting language because it runs on your computer, which means your computer has to all the processing of the language. This could mean long load times as your computer tries to interpret web pages that are loaded with JavaScript.

However, the best part about PHP is we can use it to create dynamic web pages. Whoa… another fancy technical phrase! Not really, all that means is that we can create web pages that can change their content on the fly. Since the roots of PHP lie in the web, PHP and HTML mix together very well which is important. If you’ve ever tried a different scripting language, say Perl or JavaScript, you’ll know that HTML can be hard to mix in. Using these other languages makes it difficult to enhance web design (not to say that it can’t be done, of course. Many people hold high paying jobs for using these languages on a daily basis).

PHP & the Berglund Journal

Before we get in too deep, let’s take a look at how this site is designed. The site is split up into four basic parts. First of all, you may notice that there are two menus, one on the left and one on the right. The left one is the main navigation system, and the right is the navigation system for the current issue. There is also a header area with the title graphics. The last part is the content of the page.

Okay, now we see how the page is divided up. The next step is to think about how something like this would be coded. However, there really isn’t that much to think about if we can make things modular. Making components of a web page work independently from one another makes things easier for the developer and cleaner for the user. Why is this? Well, if we make things modular, we can work on one part of the page and not have to worry about breaking other parts. This helps when we are trying to track down a particular bug or if we are experimenting with something new and don’t want everything else to “go down.” Also, by building these components, the user has a clearly defined idea of what they are supposed to do and how to use the site. The site is also more aesthetically pleasing.

All right, so now we have an idea of what we want to accomplish. We want modular pieces that

fit together to make a page. Each part should have its own function, which should be coded or constructed without relying on other parts.

Baby Step, Baby Steps…

So you may be wondering, "Where does PHP come into play?" We will get there shortly, but just a few more steps before we get up and running.

The first step is to think of a design. You may already have one. If so great, you are way ahead of the game. Just make sure that your design is nice and modular. It will save you a lot of trouble down the line. If you don't have a design, have a meeting with your team, boss, IT person, work-study student, etc. Mock up something on paper before you get to implementing.

The next step is to determine how the HTML will be coded. Before you run shrieking into the woods because you heard the word "coded," remember that HTML is in the same sentence. Since the World Wide Web is one of the most popular forms of communication today, people have made a lot of money by making the construction of web pages easier. In fact, most can get away with little or no coding at all. At the Berglund Center, (and Pacific University) we use Adobe GoLive as a standard HTML editor to build web pages. GoLive is great because it's a WYSIWYG (what you see is what you get) editor but it leaves you the option to hand code if you like. Even better, you can code PHP tags with it. However, just because I use GoLive doesn't mean you have to. Many people use Macromedia DreamWeaver, which is just as good as its Adobe counterpart. Of course you could always use a text editor, but unless you want to learn HTML too or are already proficient, this isn't the best choice.

So now we have the design and the method of implementation. Hmmm… we're forgetting something… ah, PHP! I'm assuming that either you have PHP installed on your server already or can have someone install it for you. It is a painless installation on your server if it isn't present and PHP is supported on almost all platforms (Windows, Unix, Linux, BSD, Mac OS X). I am also going to assume that you probably don't have "root" access (server administrator access) to your server. Ask your IT people to have it installed on the server that your pages will be hosted on. One more assumption: I'm assuming you have some form of web server access, whether it be apple sharing to a network drive, FTPing into a web directory, etc. Basically all I'm saying is that you have the ability to put web pages somewhere where other people can see them over the WWW. I guess if that weren't the case, you probably wouldn't be here.

Whew, now we are ready to begin working with PHP…

Implementing the Site

Now we want to begin our implementation. The first step is to actually design the "Web Template." The template is what we want our site to look like. For the sake of simplicity, we are going to use only one template throughout the entire site (like this journal) but you could easily make more than one template to accommodate various sections or departments of your website. All you would have to do is repeat the steps we will go over below.

Okay, this Journal site may be more technical than your project but we will use it as our example anyway. The first thing we did was layout tables (in GoLive) to fit our modules. Each "cell" of the table contains a piece of the interface. This allows us to be very modular. Beforehand we designed graphics using Adobe Photoshop. The rollovers were an extra added touch made with a combination of Adobe GoLive, Adobe Image Ready, and Adobe Photoshop (noticing a trend here?).

We knew that we wanted the user to be able to go to any article from any page. This meant having a menu containing all the pieces for the month. This is an important part of why we used PHP, but we will get to that later. Just keep that in mind.

Okay, so now that we've put our countless hours in and after wiping sweat off the keyboard and mouse, we have our template for the site. The content is the most important part of our website which is why everything is centered around it. That is probably what you should do with your site but once again, you may want something totally different.

What is the next step? We want to split the page into two "halves." Don't physically do it yet, let's think about it for a second. Why are we doing this? Because our content is in the middle part of the page, and we know the interface is going to be the same throughout the site, logically the only thing that is going to change often is the content. Sure, the interface has a sort of content as well, the content being the links that change monthly that link to the articles. We will get to that later. So really the only dynamic part of the page is the content. What we want to do is make a PHP sandwich around our content.

Splitting it Up

Now we have an idea of what we want to do. To accomplish this, we want to construct every page as an HTML page with only the content inside. This means (for your HTML gurus out there) that we don't want any HTML tags before the content, such as the <HTML>, <HEAD>, <BODY>, etc. tags. After the content, we don't want any of those same tags. Why? Because we are using PHP to build our interface, we are going to include PHP tags at the top and bottom of every page to fill in the interface around every page of content. Confused yet? Remember that PHP is a server-side scripting language. So when your page loads in a browser, the server sees the PHP tags in the requested web page and activates its PHP engine. The engine chugs along (quite quickly actually) and interprets your PHP tags and turns them into what you specified. In our case, we will make the tags so that is constructs our interface.

The next step in devising our PHP scheme is to actually split the page. To do this we are going to need to access the actual code of the page. Once we can see the raw HTML we can actually split the page. The first "half" should be all the HTML until we hit any content. The last "half" will contain every HTML tag after the content location. What this means is once you have your template completed, view the "code" of your project and find where the content begins. Copy the entire section of code before the content into another file and save it. Call it "front.inc." Do the same for the last half (everything after the content), but call it "end.inc." The name doesn't

really matter really; it just makes things easier for the developer. Save these files in an interface directory at the top most level of your working directory.

Now we have our interface files. The next step is to make a page with content. If you are using GoLive, you could just fill a blank page with your content. Next, we need to make the sandwich.

Go to the "code" of your content page and cut all tags out that lead up to and follow the content, not including tags that make the content work. For example, if you had a <B> tag right before the content, most likely it is a tag for the content. Leave it alone as you don't want to mess up your formatting. The same goes for end tags.

Next, we are going to make the PHP tags. To use PHP tags, the first step is to label any web page that will or does contain PHP tags as a ".php" file. This means that if we had an "index.html" file and we wanted to PHPify it, we would make it "index.php." This is so the PHP engine knows which files to process. The next step is to add the tag to the page. At first we will do this by code but once you have made one tag, you can drop back into WYSIWYG mode and edit the tags from there.

By now we should have trimmed all the beginning and end HTML code from our page. When looking at your code, we are going to add in our first PHP tag above the content. The Berglund PHP tags look something like:

<? require $_SERVER['DOCUMENT_ROOT'].'/journal/interface/front.inc'; ?>

Notice that all PHP tags start with "<?" And end with "?>". This is true for all PHP tags and be sure yours are the same. Also notice the long directory path. Guaranteed it will be different on your server. To find out what yours is, ask your server administrator the path to your working directory on your server. Note that this isn't the URL path (only the last half is) but the directory path on the file system of the server. PHP needs to know where files are located on the server. PHP is a server side scripting language after all.

Once you have everything up to your working path (ours is extra/httpd/html) the rest is filled in by you. So in our example, we have access to everything in the /html/ directory. This means that we've created a journal folder within the html folder, then an interface folder to store our interface files within the journal folder. By making the PHP tag above, we are telling PHP to include the file located at the directory path to the top of our page. Now you might see where I'm going with this… if every page has this tag at the top, and every page loads this same file, then all of our pages can use the same interface! Even better, every page can be updated by making changes to those two interface files. This is an important principle since we use it for the rest of the site.

The next obvious step is to finish by adding a tag that looks like:

<? include(extra/httpd/html/journal/interface/end.inc ?>

to the bottoms of the page. If we decided to load this up in a browser (remember that for PHP to work, the file must be located on your server with PHP installed. Simply viewing the file from your local machine isn't enough), and everything went as planned, your page should look like your template. If not, go back and make sure there are no errors with the PHP tags and make sure you did cut out any HTML tags in the wrong places. This part may take some tinkering.

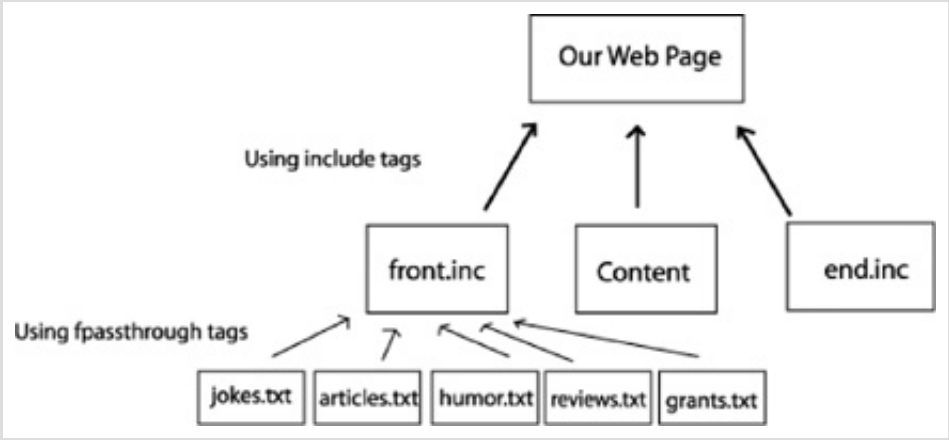
From now on, this first page will be your real template. The first template let you design the page, but the new page with the PHP tags should be the one that is reused for other pages containing content. This way, all pages will use the PHP tags you have created to pull in the interface files.

Where to go from here

Another method you can try is use the tag:

<?fpassthru(fopen("/your/directory/structure/html/journal/current/current.txt", "r"));?>

What this does is tell PHP to pass through any file that we give to the browser as HTML. This means we can build our own HTML files and have the server pass them through the web page and render them for the user. The "fopen" portion of the tag tells the server to open the file first, which it needs to do before it can be passed through. This journal uses this method to "pass through" the menu you see on the right and the Bits and Bites column. If you wanted to do this, it would give you two levels of dynamic content: The first level being the interface that would rarely change, the second level being the monthly changes. The set up is basically the same but you will have to do some back tracking of your template. The reason is you will want to include the fpassthrough tags into your interface (front.inc and end.inc) files. The only thing you will need to change is the two interface files though. The diagram below might give you a better idea of what is going on.



Above: Diagram of how each page is contructed.

Essentially, at the lowest level we are building our menus. In the far right table, we use a series of fpassthrus that link to various files each containing HTML that has links to the various articles. These are built into "front.inc." Next, the content is sandwiched with "front.inc" (which now

contains our menu for the month) and “end.inc” using include tags. All of this put together makes our tasty sandwich… err, web page.

Note that the fpassthrus use the same method of finding files as includes do. You need to specify where files are on the server’s file system, not just the URL path.

So now we can build dynamic menus within our dynamic interface. Making changes have never been easier.

Conclusion

With these two simple PHP commands, you could potentially create a dynamic site that is nearly maintenance free. I can remember not too long ago when people wanted to change the interface face of their entire site, they would run their site through some uber-program that would find the parts of the interface they would want to replace and have the program replace it. This was very time and energy inefficient. Worse yet, places would use an intern or work-study student to go through the entire site by hand and manually replace the interface on every page. Imagine doing this for a site like www.cnn.com. Now we alter just a few files and the entire site is updated, versus modifying every page in the site.

I’ll admit, this is a simplified way of doing things. However, you could take this document and use it as a starting point. Learn a little more about PHP and scripting languages in general (or HTML if you’re just beginning) and you will be amazed at what you can accomplish.

Web Resources

http://www.php.net – The main site for PHP. Contains listing of all functions with examples and documentation.
http://www.phpbuilder.net – A Great site for getting code snippets, reading articles about PHP or getting help.
http://www.devshed.com/ – Great collection of how-to’s and other articles for PHP scripting.
http://www.webmasterbase.com/ – A little more advanced but contains great how-to’s for web development.

Books:

Atkinson, Leon. Core PHP Programming. New Jersey: Prentice Hall PTR, 2001.
Zandstra, Matt. Teach Yourself PHP in 24 Hours. Indianapolis, Indiana: Sams Publishing, 2000.

This entry was posted in Uncategorized by **Editor**. Bookmark the **permalink [http://bcis.pacificu.edu/interface/?p=2260]** .

6 THOUGHTS ON “USING PHP FOR WEBSITES: CREATING A DYNAMIC INTERFACE IN THEORY AND PRACTICE”

Cleopatra Glessing
on **January 30, 2014 at 6:17 PM** said:

I really appreciate this post. I have been looking all over for this! Thank goodness I found it on Bing. You have made my day! Thx again

roulette92
on **January 31, 2014 at 12:46 AM** said:

лягушки игровые автоматы, интернет казино рулетка яндекс деньги отзывы!

community
on **February 4, 2014 at 10:20 AM** said:

I am always browsing on the internet for content articles that can benefit me. Thx!

nigeria entertainment news
on **February 4, 2014 at 10:31 AM** said:

A individual basically help to generate a lot articles or blog posts I would state. This can be the first time I frequented your internet site post and as much as now? I amazed in the analysis you produced to build this real write-up extraordinary. Good task!

nigeria entertainment news
on **February 4, 2014 at 10:41 AM** said:

good work, i adore reading your post. Maintain the good work.

php outsourcing
on **February 4, 2014 at 10:49 PM** said:

Hello! This is my first visit to your blog! We are a collection of volunteers and starting a new initiative in a community in the same niche.
Your blog provided us useful information to work on.
You have done a wonderful job!